

Assignment 2: Policy Gradient

Andrew ID: Write your Andrew ID here.

Collaborators: Write the Andrew IDs of your collaborators here (if any).

NOTE: Please do **NOT** change the sizes of the answer blocks or plots.

5 Small-Scale Experiments

5.1 Experiment 1 (Cartpole) – [5 points total]

5.1.1 Configurations

Q5.1.1

```
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 150 -b 1500 \
-dsa --exp_name q1_sb_no_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 150 -b 1500 \
-rtg -dsa --exp_name q1_sb_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 150 -b 1500 \
-rtg --exp_name q1_sb_rtg_na

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 150 -b 6000 \
-dsa --exp_name q1_lb_no_rtg_dsa

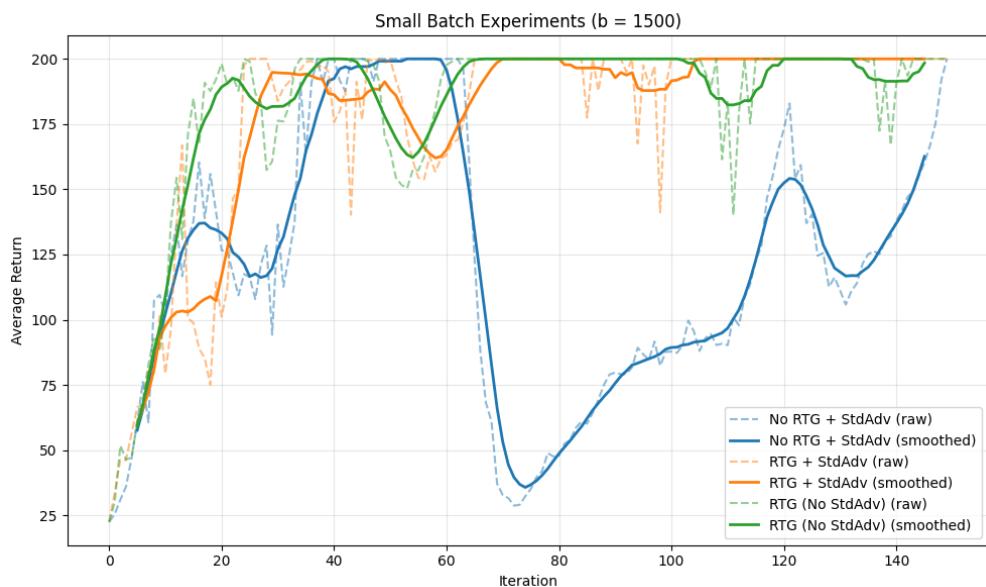
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 150 -b 6000 \
-rtg -dsa --exp_name q1_lb_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 150 -b 6000 \
-rtg --exp_name q1_lb_rtg_na
```

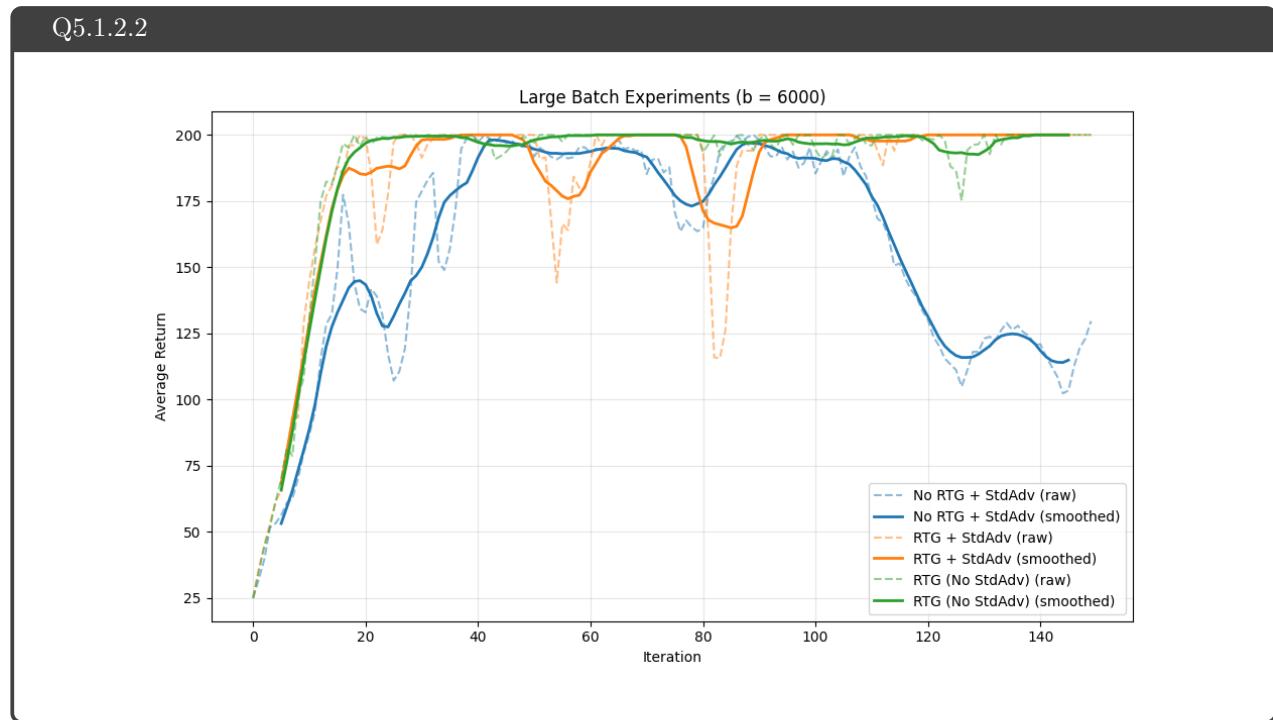
5.1.2 Plots

5.1.2.1 Small batch – [1 points]

Q5.1.2.1

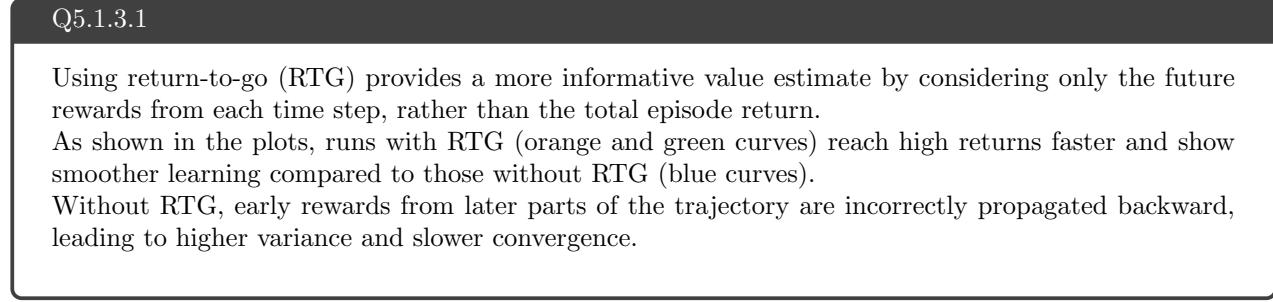


5.1.2.2 Large batch – [1 points]

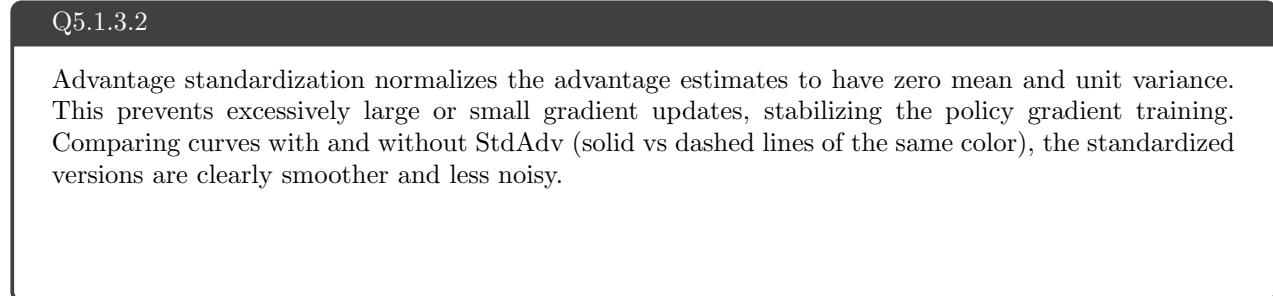


5.1.3 Analysis

5.1.3.1 Value estimator – [1 points]



5.1.3.2 Advantage standardization – [1 points]



5.1.3.3 Batch size – [1 points]

Q5.1.3.3

Larger batch sizes lead to more stable gradient estimates because they average over more trajectories. Comparing the two figures, the large-batch experiment ($b=6000$) produces much smoother learning curves with smaller fluctuations and reaches optimal performance more reliably. However, the smaller batch ($b=1500$) learns faster at the beginning but is more unstable and sometimes dips in performance.

5.2 Experiment 2 (InvertedPendulum) – [4 points total]

5.2.1 Configurations – [1.5 points]

Q5.2.1

```
python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 \
--ep_len 1000 --discount 0.92 -n 100 -l 2 -s 64 -b <b*> -lr <r*> -rtg \
--exp_name q2_b<b*>_r<r*>

python -m rob831.scripts.run_hw2 --env_name InvertedPendulum-v4 \
--ep_len 1000 --discount 0.92 \
-n 100 -l 2 -s 64 -b 1000 -lr 0.005 -rtg --exp_name q2_b1000_r0.005

python -m rob831.scripts.run_hw2 --env_name InvertedPendulum-v4 \
--ep_len 1000 --discount 0.92 \
-n 100 -l 2 -s 64 -b 5000 -lr 0.02 -rtg --exp_name q2_b5000_r0.02

python -m rob831.scripts.run_hw2 --env_name InvertedPendulum-v4 \
--ep_len 1000 --discount 0.92 \
-n 100 -l 2 -s 64 -b 10000 -lr 0.03 -rtg --exp_name q2_b10000_r0.03
```

5.2.2 smallest b* and largest r* (same run) – [1.5 points]

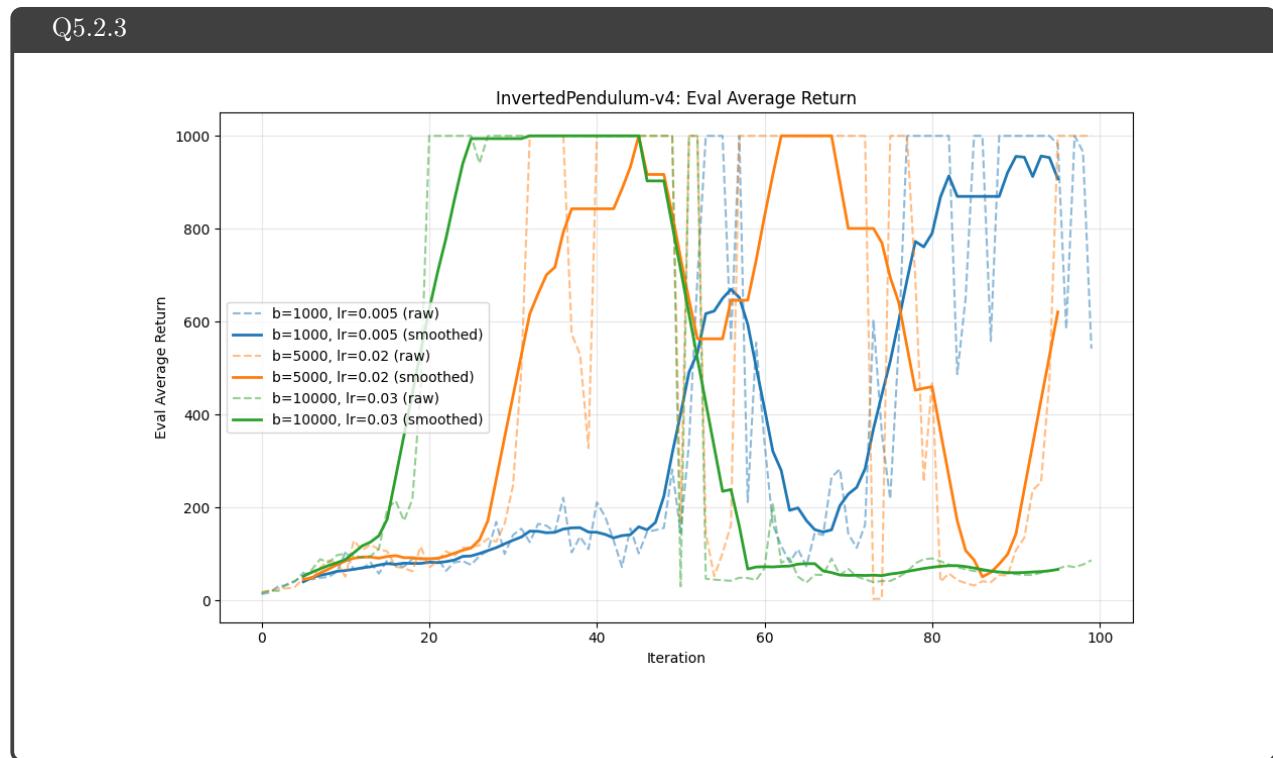
Q5.2.2

The smallest batch size and largest learning rate combination (same run) that achieves the optimum (1000) within 100 iterations is:

$$b^* = 1000 \quad r^* = 0.005$$

Although larger learning rates led to faster early improvements, they caused instability and performance drops. The configuration with $b=1000$ and $lr=0.005$ eventually reached the maximum score after about 80 iterations, while maintaining stable learning behavior.

5.2.3 Plot – [1 points]



7 More Complex Experiments

7.1 Experiment 3 (LunarLander) – [1 points total]

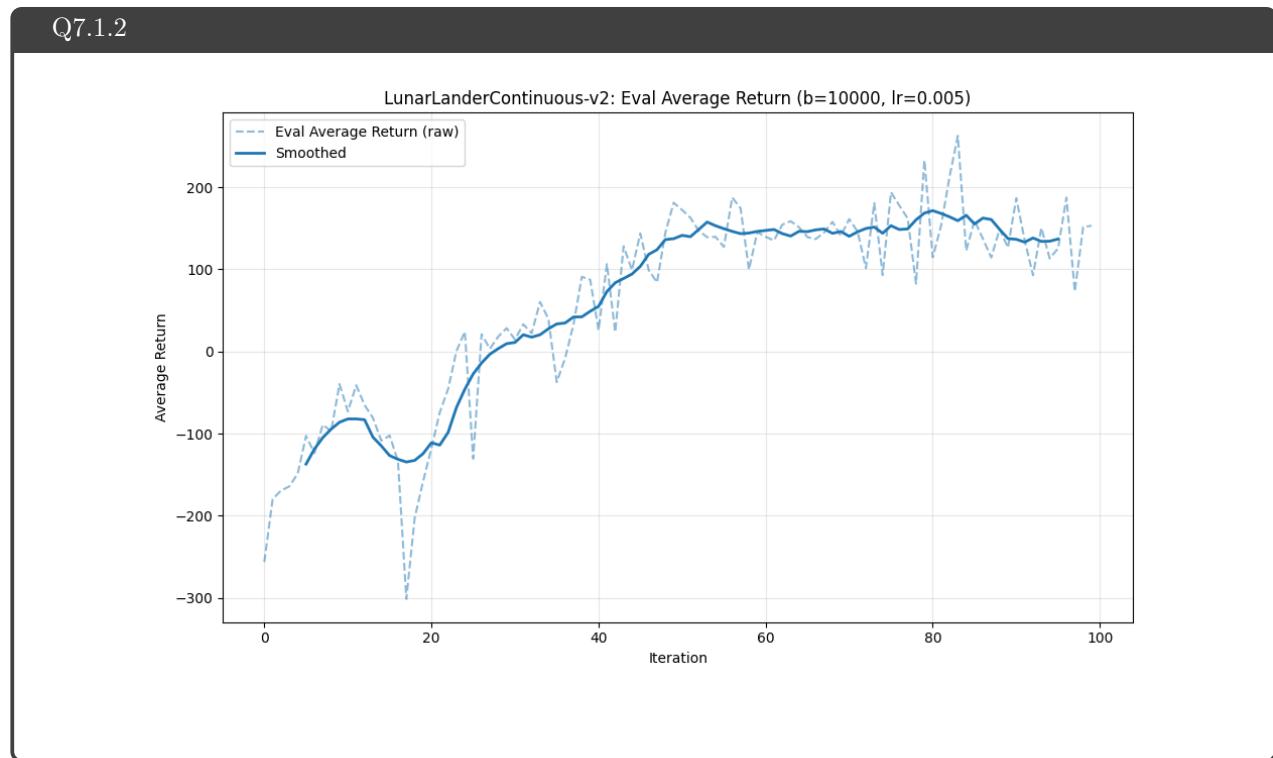
7.1.1 Configurations

Q7.1.1

```
python rob831/scripts/run_hw2.py \
--env_name LunarLanderContinuous-v4 --ep_len 1000 \
--discount 0.99 -n 100 -l 2 -s 64 -b 10000 -lr 0.005 \
--reward_to_go --nn_baseline --exp_name q3_b10000_r0.005

python -m rob831.scripts.run_hw2 \
--env_name LunarLanderContinuous-v2 --ep_len 1000 \
--discount 0.99 -n 100 -l 2 -s 64 -b 10000 -lr 0.005 \
--reward_to_go --nn_baseline --exp_name q3_b10000_r0.005
```

7.1.2 Plot – [1 points]



7.2 Experiment 4 (HalfCheetah) – [1 points]

7.2.1 Configurations

Q7.2.1

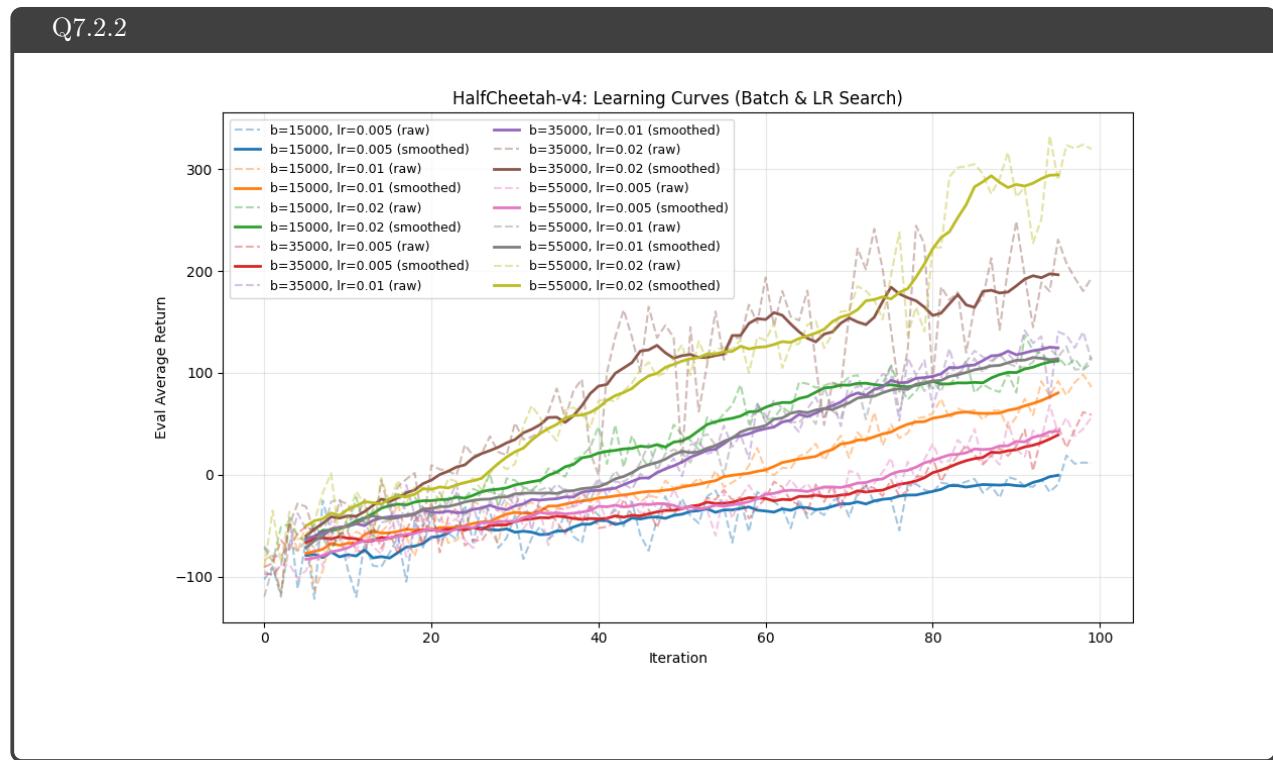
```
python -m rob831.scripts.run_hw2 --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 15000 -lr 0.005 --rtg --nn_baseline \
--exp_name q4_search_b15000_lr0.005_rtg_nnbaseline

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg \
--exp_name q4_search_b10000_lr0.02_rtg

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 --nn_baseline \
--exp_name q4_search_b10000_lr0.02_nnbaseline

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_search_b10000_lr0.02_rtg_nnbaseline
```

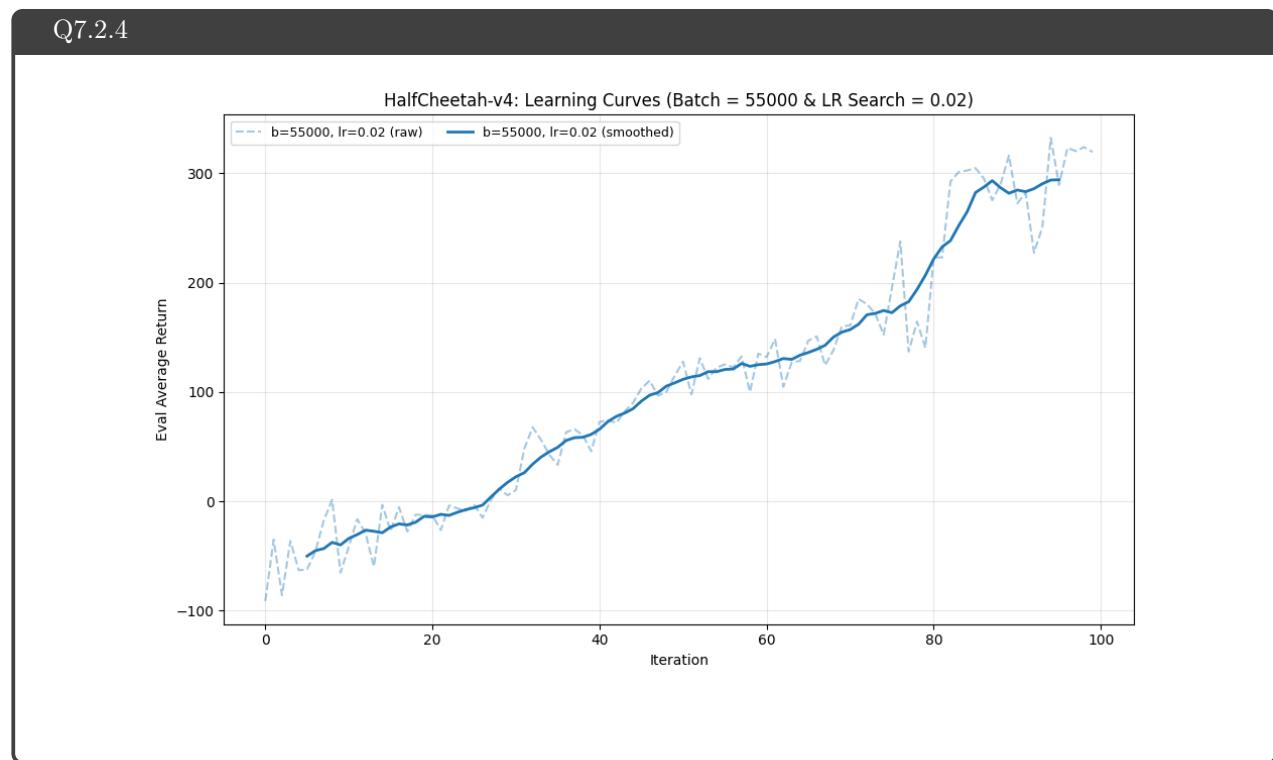
7.2.2 Plot – [1 points]



7.2.3 Optimal b^* and r^* – [0.5 points]

Q7.2.3

The best performing configuration is batch size $b^* = 55,000$ and learning rate $r^* = 0.02$. This setup achieves the highest evaluation return (300) and the fastest convergence among all tested combinations.

7.2.4 Plot – [0.5 points]**7.2.5 Describe how b^* and r^* affect task performance – [0.5 points]**

Q7.2.5

Increasing the batch size b improves training stability and reduces gradient variance, leading to smoother learning and better performance.
A higher learning rate r accelerates convergence, but if too large, it causes instability.

7.2.6 Configurations with optimal b^* and r^* – [0.5 points]

Q7.2.6

```
python -m rob831.scripts.run_hw2 --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 55000 -lr 0.02 \
--exp_name q4_b55000_r0.02

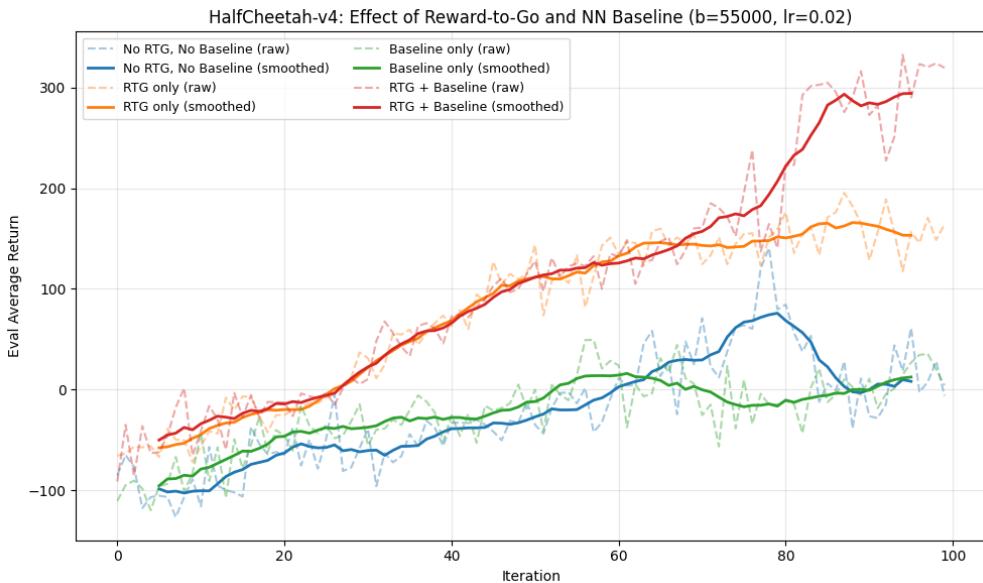
python -m rob831.scripts.run_hw2 --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 55000 -lr 0.02 -rtg \
--exp_name q4_b55000_r0.02_rtg

python -m rob831.scripts.run_hw2 --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 55000 -lr 0.02 --nn_baseline \
--exp_name q4_b55000_r0.02_mnbaseline

python -m rob831.scripts.run_hw2 --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 55000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_b55000_r0.02_rtg_nnb
```

7.2.7 Plot for four runs with optimal b^* and r^* – [0.5 points]

Q7.2.7



8 Implementing Generalized Advantage Estimation

8.1 Experiment 5 (Hopper) – [4 points]

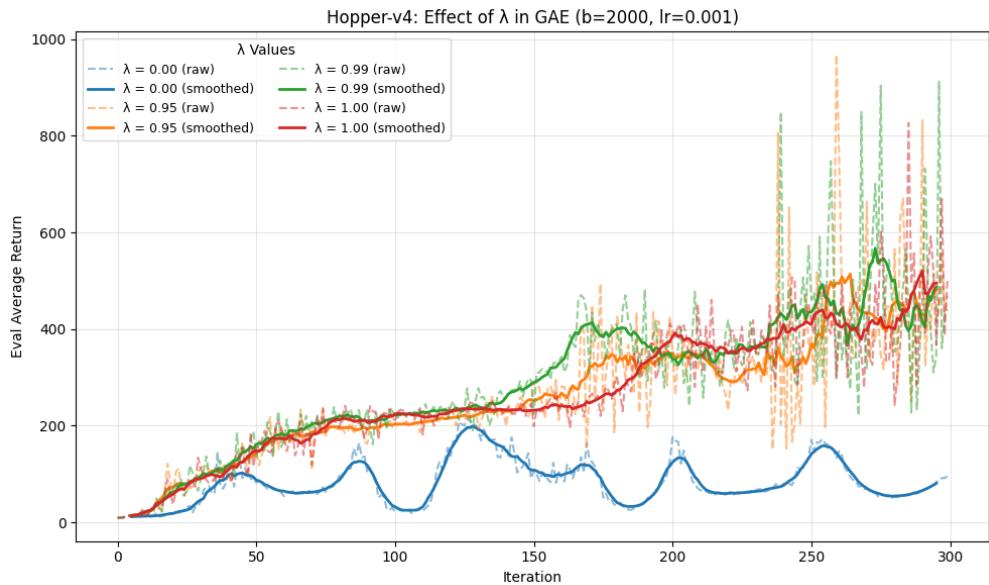
8.1.1 Configurations

Q8.1.1

```
# λ ∈ [0, 0.95, 0.99, 1]
python rob831/scripts/run_hw2.py \
--env_name Hopper-v4 --ep_len 1000 \
--discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 \
--reward_to_go --nn_baseline --action_noise_std 0.5 --gae_lambda <λ> \
--exp_name q5_b2000_r0.001_lambda<λ>
```

8.1.2 Plot – [2 points]

Q8.1.2



8.1.3 Describe how λ affects task performance – [2 points]

Q8.1.3

Small λ (e.g., 0) → higher bias, low variance → learning is slower and less effective.

Large λ (e.g., 0.95–1.0) → lower bias, higher variance → learning is faster and achieves higher returns, though possibly more noisy.

$\lambda = 0.95\text{--}0.99$ yields the best overall performance, balancing learning speed and stability.